

The Random Number Generator Validation System (RNGVS)

May 25, 2004

Lawrence E. Bassham III

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1	INTRODUCTION.....	1
2	SCOPE.....	1
3	CONFORMANCE.....	2
4	DEFINITIONS AND ABBREVIATIONS	2
4.1	DEFINITIONS.....	2
4.2	ABBREVIATIONS	2
5	DESIGN PHILOSOPHY OF THE RANDOM NUMBER GENERATOR VALIDATION SYSTEM	3
6	RNGVS TESTS	3
6.1	CONFIGURATION INFORMATION	4
6.2	THE VARIABLE SEED TEST	5
6.3	THE MONTE CARLO TEST	6
6.4	SPECIFIC REQUIREMENTS FOR TESTING THE RNG FOUND IN ANSI X9.31 APPENDIX A.2.4.....	7
APPENDIX A	REFERENCES	8
APPENDIX B	EXAMPLES OF <i>REQUEST</i>, <i>FAX</i>, <i>RESPONSE</i>, AND <i>SAMPLE</i> FILES	9
B.1	EXAMPLES OF <i>REQUEST</i> FILES.....	9
B.1.1	FIPS186_VST.req.....	9
B.1.2	ANSI962_VST.req	10
B.1.3	ANSI931_VST.req	12
B.1.4	FIPS186_MCT.req	13
B.1.5	ANSI962_MCT.req.....	14
B.1.6	ANSI931_MCT.req.....	15
B.2	EXAMPLES OF <i>FAX</i> FILES.....	16
B.2.1	FIPS186_VST.fax.....	16
B.2.2	ANSI962_VST.fax.....	18
B.2.3	ANSI931_VST.fax.....	20
B.2.4	FIPS186_MCT.fax.....	21
B.2.5	ANSI962_MCT.fax.....	22
B.2.6	ANSI931_MCT.fax.....	23
B.3	EXAMPLES OF <i>RESPONSE</i> FILES.....	23
B.3.1	FIPS186_VST.rsp.....	23
B.3.2	ANSI962_VST.rsp	25
B.3.3	ANSI931_VST.rsp	27
B.3.4	FIPS186_MCT.rsp	28
B.3.5	ANSI962_MCT.rsp.....	29
B.3.6	ANSI931_MCT.rsp.....	31
B.4	EXAMPLES OF <i>SAMPLE</i> FILES.....	31
B.4.1	FIPS186_VST.sam.....	31
B.4.2	ANSI962_VST.sam.....	33
B.4.3	ANSI931_VST.sam.....	35

B.4.4	FIPS186_MCT.sam.....	36
B.4.5	ANSI962_MCT.sam.....	37
B.4.6	ANSI931_MCT.sam.....	38

1 Introduction

This document, *The Random Number Generation Validation System (RNGVS)* specifies the procedures involved in validating implementations of the various Random Number Generators (RNG) as specified and approved in FIPS 186-2, *Digital Signature Standard (DSS)* [1]; ANSI X9.62-1998, *Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA)* [2]; and ANSI X9.31-1998, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)* [3]. The RNGVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the RNGVS. Included are the specifications for testing the individual RNG algorithms from the approved standards implemented by the IUT. These algorithms are:

- DSA – FIPS 186-2 Appendix 3.1 – Algorithm for Computing m values of x (using SHA-1 and/or DEA),
- DSA – FIPS 186-2 Appendix 3.2 – Algorithm for Precomputing One or More k and r Values (using SHA-1 and/or DEA),
- ECDSA – ANSI X9.62-1998 Appendix A.4 - Pseudorandom Number Generation (using SHA-1 and/or DEA), and
- rDSA – ANSI X9.31-1998 Appendix A.2.4 - Generating Pseudo Random Numbers Using the DEA.

This document defines the purpose, the design philosophy, and the high-level description of the validation process for various RNGs. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of an RNG are presented. The requirements described include a specification of the data communicated between the IUT and the RNGVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the RNGVS. Additionally, an appendix is also provided containing samples of input and output files for the RNGVS.

2 Scope

This document specifies the tests required to validate IUTs for conformance to RNGs specified in various standards [1][2][3]. When applied to IUTs that implement an RNG, the RNGVS provides testing to determine the correctness of the RNGs contained in the implementation. The RNGVS is composed of two tests for each RNG algorithm implemented: the Variable Seed Test (VST) and the Monte Carlo Test (MCT). In addition to determining conformance to the cryptographic specifications, the RNGVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the RNG implementation.

3 Conformance

The successful completion of the tests contained within the RNGVS is required to be validated as conforming to the particular RNG standard. Testing for the cryptographic module in which the RNG is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*.[4]

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING
CMT laboratory	Cryptographic Module Testing laboratory that operates the RNGVS
Digital Signature Algorithm	The algorithm specified in FIPS 186-2, <i>Digital Signature Algorithm (DSA)</i>
Elliptic Curve Digital Signature Algorithm	The algorithm specified in ANSI X9.62-1998, <i>Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA)</i>
Reversible Digital Signature Algorithm	The algorithm specified in ANSI X9.31-1998, <i>Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)</i>
Secure Hash Algorithm	The algorithm specified in FIPS 180-2, <i>Secure Hash Standard (SHS)</i>

4.2 Abbreviations

ABBREVIATION	MEANING
DEA	Data Encryption Algorithm
DSA	Digital Signature Algorithm specified in FIPS 186-2
ECDSA	Elliptic Curve Digital Signature Algorithm specified in ANSI X9.62-1998
FIPS	Federal Information Processing Standard
IUT	Implementation Under Test
MCT	Monte Carlo Test
rDSA	Reversible Digital Signature Algorithm specified in ANSI X9.31-1998

RNG	Random Number Generator
RNGVS	Random Number Generator Validation System
SHA-1	Secure Hash Algorithm-1 specified in FIPS 180-2
VST	Variable Seed Test

5 Design Philosophy Of The Random Number Generator Validation System

The RNGVS is designed to test conformance to the various approved RNG specifications rather than provide a measure of a product's security. The validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation should not be interpreted as an evaluation or endorsement of overall product security.

The RNGVS has the following design philosophy:

1. The RNGVS is designed to allow the testing of an IUT at locations remote to the RNGVS. The RNGVS and the IUT communicate data via *REQUEST* and *RESPONSE* files. The RNGVS also generates *SAMPLE* files to provide the IUT with a sample of what the *RESPONSE* file should look like.
2. The testing performed within the RNGVS utilizes statistical sampling (i.e., only a small number of the possible cases are tested); hence, the successful validation of a device does not imply 100% conformance with the standard.

6 RNGVS Tests

The RNGVS tests various RNG algorithm implementations for their conformance to their respective standards. The testing for each algorithm consists of two tests: the Variable Seed Test (VST) and a Monte Carlo Test (MCT). The algorithms tested are:

- DSA – FIPS 186-2 Appendix 3.1 – Algorithm for Computing m values of x (using SHA-1 and/or DEA),
- DSA – FIPS 186-2 Appendix 3.2 – Algorithm for Precomputing One or More k and r Values (using SHA-1 and/or DEA),

- ECDSA – ANSI X9.62-1998 Appendix A.4 - Pseudorandom Number Generation (using SHA-1 and/or DEA), and
- rDSA – ANSI X9.31-1998 - Generating Pseudo Random Numbers Using the DEA.

6.1 Configuration Information

To initiate the validation process of the RNGVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of an RNG. The vendor's implementation is referred to as the Implementation Under Test (IUT). The request for validation includes background information describing the IUT along with information needed by the RNGVS to perform the specific tests. More specifically, the request for validation includes:

1. Vendor Name;
2. Product Name;
3. Product Version;
4. Implementation in software, firmware, or hardware;
5. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;
6. Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences); and
7. Specific configuration information, as needed, for the RNG algorithms implemented by the IUT:
 - a) DSA - Generation of X
 - Whether the implementation contains the original specification, the Change Notice (10/5/2001) specification, or both;
 - Whether the G function is constructed from SHA-1, DEA, or both;
 - Whether the algorithm is used as specified or as a generic purpose random number generator¹, or both;

¹ As per the last section of the Change Notice 1 of FIPS 186-2, dated 2001 October 5, the algorithm specified in Appendix 3.1 of FIPS 186-2 or algorithm 1 of the Change Notice can be used as a general purpose RNG. In this case the “mod q” at the end of the algorithms is omitted. The file names associated with these test have “GEN” preceding the extension.

- The minimum and maximum seed lengths ($160 \leq \text{minlen} \leq \text{ maxlen} \leq 512$) the implementation supports; and
 - The value of the domain parameter Q if a specific value is required by the implementation.
- b) DSA - Generation of K
- Whether the implementation contains the original specification, the Change Notice (10/5/2001) specification, or both;
 - Whether the G function is constructed from SHA-1, DEA, or both;
 - The minimum and maximum seed lengths ($160 \leq \text{minlen} \leq \text{ maxlen} \leq 512$) the implementation supports; and
 - The value of the domain parameter Q if a specific value is required by the implementation.
- c) ECDSA - Pseudorandom Number Generation
- The specific NIST Recommended Curves the implementation supports;
 - Whether the G function is constructed from SHA-1, DEA, or both; and
 - The minimum and maximum seed lengths ($160 \leq \text{minlen} \leq \text{ maxlen} \leq 512$) the implementation supports.

6.2 The Variable Seed Test

The Variable Seed Test provides a series of seeds, each with a one-bit difference from the previous seed. The algorithm uses the *SEED* value to generate the random value with the algorithm being tested.

The RNGVS:

A. Creates a *REQUEST* file (Filename: <Alg>_VST[GEN].req) containing:

1. The Product Name;
2. The algorithm being tested; and
3. The *SEED* values used as input to the RNG algorithm.

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

B. Creates a *FAX* file (Filename: <Alg>_VST[GEN].fax) containing:

1. The Product Name;

2. The algorithm being tested;
3. The *SEED* values used as input to the RNG algorithm; and
4. The random values produced from the *SEED* provided.

Note: The CMT laboratory retains the *FAX* file.

The IUT:

- A. Generates the requested random values from the *SEED*'s specified in the *REQUEST* file.
- B. Creates a *RESPONSE* file (Filename: <Alg>_VST[GEN].rsp) containing:
 1. The Product Name;
 2. The algorithm being tested;
 3. The *SEED* values used as input to the RNG algorithm; and
 4. The random values produced from the *SEED* provided.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the RNGVS.

The RNGVS:

- A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.
- B. If all values match, records PASS for this test; otherwise, records FAIL.

6.3 The Monte Carlo Test

The Monte Carlo Test provides a *SEED* value to be used by the algorithm being tested. The algorithm uses the *SEED* value to generate a sequence of random value with the algorithm being tested. The algorithm being tested prescribes a method for creating a new *SEED* value from the existing value.

For each *SEED* value supplied, the IUT produces a sequence of 10,000 random values. The final random value is supplied in the *RESPONSE* file for verification.

The RNGVS:

- A. Creates a *REQUEST* file (Filename: <Alg>_MCT[GEN].req) containing:
 1. The Product Name;
 2. The algorithm being tested; and
 3. One or more *SEED* values (depending on the algorithm being tested).

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

- B. Creates a *FAX* file (Filename: <Alg>_MCT[GEN].fax) containing:

1. The information from the *REQUEST* file; and
2. For each *SEED* value present, the 10,000th random value derived from the RNG algorithm being tested.

Note: The CMT laboratory retains the *FAX* file.

The IUT:

- A. For each *SEED* value found in the *REQUEST* file, generates 10,000 random values.
- B. Creates a *RESPONSE* file (Filename: <Alg>_MCT[GEN].rsp) containing:
 1. The information from the *REQUEST* file; and
 2. For each *SEED* value, the 10,000th random value it produces.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the RNGVS.

The RNGVS:

- A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.
- B. If the results for all *SEED* values match, records PASS for this test; otherwise, records FAIL.

6.4 Specific Requirements for Testing the RNG found in ANSI X9.31 Appendix A.2.4

The *SEED* value specified in the VST and the MCT are used as the input vector V of ANSI X9.31-1998 Appendix A.2.4. The initial date/time vector and the two key values are supplied by the RNGVS. In the Monte Carlo Test, subsequent date/time vectors are calculated by incrementing the previous value by one.

Appendix A References

- [1] *Digital Signature Standard (DSS)*, FIPS Publication 186-2 (+Change Notice), National Institute of Standards and Technology, January 2000.
- [2] *Public Key Cryptography for Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI X9.62-1988, January 1999.
- [3] *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, ANSI X9.31-1988, September 1998.
- [4] *Security Requirements for Cryptographic Modules*, FIPS Publication 140-2, National Institute of Standards and Technology, May 2001.

Appendix B Examples of *REQUEST*, *FAX*, *RESPONSE*, and *SAMPLE* Files

The following are partial examples of *REQUEST*, *FAX*, *RESPONSE*, and *SAMPLE* files for each of the algorithms tested by the RNGVS.

B.1 Examples of *REQUEST* Files

B.1.1 FIPS186_VST.req

```
# CAVS 2.2
# "FIPS 186" information for "Demo Product"
# Generators selected: Xorg
# G-Functions selected: SHA-1 DES
# Generated on Tue Jun 03 09:18:10 2003

[Xorg - SHA1]

Q = 9eedc3fde07ed95848e3e0f0c7e690ad1327e511

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 1
b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

...
COUNT = 158
b = 160
XKey = ffffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
```

B.1.2 ANSI962_VST.req

```
# CAVS 2.2
# "ANSI X9.62" information for "Demo Product"
# Generators selected: P-192
```

```

# G-Functions selected: SHA-1 DES
# Generated on Tue Jun 03 09:18:15 2003

[P-192 - SHA1]

N = fffffffffffffffffff99def836146bc9b1b4d22831

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 1
b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

...
COUNT = 158
b = 160
XKey = ffffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 159
b = 160
XKey = ffffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000

[P-192 - DES]

N = fffffffffffffffffff99def836146bc9b1b4d22831

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000

COUNT = 1

```

```

b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

...
COUNT = 158
b = 160
XKey = ffffffff0000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

COUNT = 159
b = 160
XKey = ffffffff0000000000000000000000000000000000000000000000000000000
XSeed = 0000000000000000000000000000000000000000000000000000000000000000

```

B.1.3 ANSI931_VST.req

```

# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

COUNT = 0
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = 8000000000000000

COUNT = 1
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3d
V = c000000000000000

COUNT = 2
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3e

```

B.1.4 FIPS186_MCT.req

B.1.5 ANSI962_MCT.req

B.1.6 ANSI931_MCT.req

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = d5538f9cf450f53c
```

B.2 Examples of *FAX* Files

B.2.1 FIPS186_VST.fax

```
# CAVS 2.2
# "FIPS 186" information for "Demo Product"
# Generators selected: Xorg
# G-Functions selected: SHA-1 DES
# Generated on Tue Jun 03 09:18:10 2003

[Xorg - SHA1]

Q = 9eedc3fde07ed95848e3e0f0c7e690ad1327e511

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 3b4bdff07dec71b4e971defecd7987e39cb021f8

COUNT = 1
b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 5dde2767380ae76c3a884ea4240feb11468729e7

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 12098135df8852d450ee60b3fe0e368eb06f18e1

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 3f87039d81ff007b02d2a4cbf1eb28be42ad9fc3

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 76aae1571999ccf26fc1d8050da716fc1d4601e

...
COUNT = 158
b = 160
XKey = fffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 089fc77cf8929c246fce00b92c27676cca08e75

COUNT = 159
```

```

b = 160
XKey = fffffffffffffffffffff
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 2fa01580d4bf0f60509990be4e084272e95a48da

[Xorg - DES]

Q = 9eedc3fde07ed95848e3e0f0c7e690ad1327e511

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 870f8a5c137c1093c7a8fd53179ff2ce7b0cf127

COUNT = 1
b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 218de164a97806592fb03f086a9c3c036f1e6e9d

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 98169623fa3daf298513ec5ca79ac3cac2950fdb

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 5d52a7751e05fd2af30bc9f2087084429cf5c11e

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 34d5f9134b28785a277229ce8e1d09c6a1a76820

...
COUNT = 158
b = 160
XKey = ffffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 14329258f79a332ae886bb9e999630b7621c31fd

COUNT = 159
b = 160
XKey = ffffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 7fc24dcfc05cf975f702ac609f34b97d219a9f93

```

B.2.2 ANSI962_VST.fax

```
# CAVS 2.2
# "ANSI X9.62" information for "Demo Product"
# Generators selected: P-192
# G-Functions selected: SHA-1 DES
# Generated on Tue Jun 03 09:18:15 2003

[P-192 - SHA1]

N = fffffffffffffffffff99def836146bc9b1b4d22831

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 7ebf6e9c0f9828de5590fe0139c27574ae95ba3c075384c9

COUNT = 1
b = 160
XKey = c00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = f1ab3e2f58395deccaf697271e81cea97971fdc6a7444b2b

COUNT = 2
b = 160
XKey = e00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = a5a8cb5f81d6efa6ba3f7b02a677cbb8338faa8f69228fc2

COUNT = 3
b = 160
XKey = f00000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 4857d8e69abc4bdd26ee8d4821e59578e65bfc47ceec9061

COUNT = 4
b = 160
XKey = f80000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = d7c9d482b774b00f48b82569e77fe80918af87c62e748ccf

...
COUNT = 158
b = 160
XKey = fffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 7a9eb9d0cef5938aa1b4c73912bc497c73a129a1099809c

COUNT = 159
b = 160
XKey = fffffffffffffffffffe
```

```

XSeed = 000000000000000000000000000000000000000000000000000000000000
X = 4e743d8ad7eb660bb44e06143344c45b7f758c2efb2c35dc

[P-192 - DES]

N = fffffffffffffffffff99def836146bc9b1b4d22831

COUNT = 0
b = 160
XKey = 80000000000000000000000000000000000000000000000000000000000000
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = f4f47f39c0922d1c8fd50855e7888eb02c3a14ac0f4d3378

COUNT = 1
b = 160
XKey = c0000000000000000000000000000000000000000000000000000000000000
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = f10e7d54e4c290833df478f2147aec18cc4b68e659ab803

COUNT = 2
b = 160
XKey = e0000000000000000000000000000000000000000000000000000000000000
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = 06b88cd5f46a23efb3a23e1b0b00670bd18316d72a7065b7

COUNT = 3
b = 160
XKey = f0000000000000000000000000000000000000000000000000000000000000
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = 14ff126a72431e664ebd6fbf2ced6ea53bff2f883924f81c

COUNT = 4
b = 160
XKey = f8000000000000000000000000000000000000000000000000000000000000
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = f385520dbb15dde89fd3787630ddb6d0724bf48e30ccd012

...
COUNT = 158
b = 160
XKey = ffffffffffffffffffffe
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = 028701bdc0b422547932d71271b4ddf9af830a95853a4dcf

COUNT = 159
b = 160
XKey = ffffffffffffffffffffe
XSeed = 00000000000000000000000000000000000000000000000000000000000000
X = 054892327a14e4b6ef7d5c97c59688055a022cda0cf20e26

```

B.2.3 ANSI931_VST.fax

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

COUNT = 0
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3c
V = 8000000000000000
R = 944dc7210d6d7fd7

COUNT = 1
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3d
V = c000000000000000
R = af1a648591bb7c2c

COUNT = 2
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3e
V = e000000000000000
R = 221839b07451e423

COUNT = 3
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3f
V = f000000000000000
R = eba9271e04043712

COUNT = 4
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f40
V = f800000000000000
R = 02433c9417a3326f

...
COUNT = 62
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f7a
V = ffffffff
R = 13eeb44dcba310f1

COUNT = 63
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
```

```
DT = c89a1d888ed12f7b  
V = ffffffff  
R = e7e2b2964f36ed41
```

B.2.4 FIPS186_MCT.fax

B.2.5 ANSI962_MCT.fax

B.2.6 ANSI931_MCT.fax

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = d5538f9cf450f53c
R = 77c695c33e51c8c0
```

B.3 Examples of *RESPONSE* Files

B.3.1 FIPS186_VST.rsp

```
# CAVS 2.2
# "FIPS 186" information for "Demo Product"
# Generators selected: Xorg
# G-Functions selected: SHA-1 DES

[Xorg - SHA1]
```

```

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 3b4bdff07dec71b4e971defecd7987e39cb021f8

COUNT = 1
b = 160
XKey = c000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 5dde2767380ae76c3a884ea4240feb11468729e7

COUNT = 2
b = 160
XKey = e000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 12098135df8852d450ee60b3fe0e368eb06f18e1

COUNT = 3
b = 160
XKey = f000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 3f87039d81ff007b02d2a4cbf1eb28be42ad9fc3

COUNT = 4
b = 160
XKey = f800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 76aae1571999ccf26fc1d8050da716fc1d4601e

...
COUNT = 158
b = 160
XKey = fffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 089fc77cf8929c246fce00b92c27676cca08e75

COUNT = 159
b = 160
XKey = fffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 2fa01580d4bf0f60509990be4e084272e95a48da

[Xorg - DES]

Q = 9eedc3fde07ed95848e3e0f0c7e690ad1327e511

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 870f8a5c137c1093c7a8fd53179ff2ce7b0cf127

```

```

COUNT = 1
b = 160
XKey = c000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 218de164a97806592fb03f086a9c3c036f1e6e9d

COUNT = 2
b = 160
XKey = e000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 98169623fa3daf298513ec5ca79ac3cac2950fdb

COUNT = 3
b = 160
XKey = f000000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 5d52a7751e05fd2af30bc9f2087084429cf5c11e

COUNT = 4
b = 160
XKey = f800000000000000000000000000000000000000000000000000000000000000
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 34d5f9134b28785a277229ce8e1d09c6a1a76820

...
COUNT = 158
b = 160
XKey = fffffffffffffffffffe
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 14329258f79a332ae886bb9e999630b7621c31fd

COUNT = 159
b = 160
XKey = fffffffffffffffffff
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = 7fc24dcfc05cf975f702ac609f34b97d219a9f93

```

B.3.2 ANSI962_VST.rsp

```

# CAVS 2.2
# "ANSI X9.62" information for "Demo Product"
# Generators selected: P-192
# G-Functions selected: SHA-1 DES

[P-192 - SHA1]

N = fffffffffffffffffff99def836146bc9b1b4d22831

COUNT = 0
b = 160
XKey = 800000000000000000000000000000000000000000000000000000000000000

```


B.3.3 ANSI931_VST.rsp

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"

COUNT = 0
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = 8000000000000000
R = 944dc7210d6d7fd7

COUNT = 1
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3d
V = c000000000000000
```

```

R = af1a648591bb7c2c

COUNT = 2
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3e
V = e0000000000000000
R = 221839b07451e423

COUNT = 3
Key1 = 32ad2932fd4c6202
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3f
V = f0000000000000000
R = eba9271e04043712

COUNT = 4
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f40
V = f800000000000000
R = 02433c9417a3326f

...
COUNT = 62
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f7a
V = ffffffff
R = 13eeb44dcba310f1

COUNT = 63
Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f7b
V = ffffffff
R = e7e2b2964f36ed41

```

B.3.4 FIPS186_MCT.rsp

```

# CAVS 2.2
# "FIPS 186" information for "Demo Product"
# Generators selected: Xorg
# G-Functions selected: SHA-1 DES

[Xorg - SHA1]

Q = 9eedc3fde07ed95848e3e0f0c7e690ad1327e511

COUNT = 0
b = 160

```

B.3.5 ANSI962_MCT.rsp

```
# CAVS 2.2
# "ANSI X9.62" information for "Demo Product"
```



```
XSeed = 0000000000000000000000000000000000000000000000000000000000000000  
X = 136af998090924a478cbe45d713ce405f9ae95485a9f7a57
```

B.3.6 ANSI931_MCT.rsp

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"

Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = d5538f9cf450f53c
R = 77c695c33e51c8c0
```

B.4 Examples of *SAMPLE* Files

B.4.1 FIPS186_VST.sam

B.4.2 ANSI962_VST.sam


```

...
COUNT = 158
b = 160
XKey = ffffffffffffffes
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = ?

COUNT = 159
b = 160
XKey = ffffffffffffffes
XSeed = 000000000000000000000000000000000000000000000000000000000000000
X = ?

```

B.4.3 ANSI931_VST.sam

```

# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

COUNT = 0
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3c
V = 8000000000000000
R = ?

COUNT = 1
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3d
V = c000000000000000
R = ?

COUNT = 2
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3e
V = e000000000000000
R = ?

COUNT = 3
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f3f
V = f000000000000000
R = ?

COUNT = 4
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f40

```

```
V = f800000000000000
R = ?

...
COUNT = 62
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f7a
V = ffffffffffffffe
R = ?

COUNT = 63
Key1 = 75c71ae5a11a232c
Key2 = 40256dc94f767b0
DT = c89a1d888ed12f7b
V = ffffffffffffffe
R = ?
```

B.4.4 FIPS186_MCT.sam

B.4.5 ANSI962_MCT.sam

B.4.6 ANSI931_MCT.sam

```
# CAVS 2.2
# "ANSI X9.31" information for "Demo Product"
# Generated on Tue Jun 03 09:18:59 2003

Key1 = 75c71ae5a11a232c
Key2 = 40256dcd94f767b0
DT = c89a1d888ed12f3c
V = d5538f9cf450f53c
R = ?
```